Eur päisches Patentamt | European Patent Office | Office européen des brevets

Q77462
1572

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

02025251.6

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 02025251.6
Demande no:

Anmeldetag:
Date of filing: 12.11.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

ALCATEL
54, rue la Boétie
75008 Paris
FRANCE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Fast restoration

In Anspruch genommene Prioriät(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

H04L/

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

Fast Restoration

Background of the Invention

The present invention relates to the field of telecommunications and more particularly to a method for re-configuring a network element of a transmission network to restore traffic after occurrence of a failure.

Background of the Invention

A Digital Cross Connect (DXC) must react in some scenarios as fast as possible. For instance in the case of a fiber cut, traffic from this fiber must be restored very quickly ("Restoration"). In meshed networks (e.g. SDH or SONET networks) the capability of fast restoration is very important. The required configuration changes shall thus be accelerated.

DXC systems typically use a layered SW architecture, where each layer checks, processes, and stores configuration requests before these requests are forwarded to the next lower layer. Acceleration of this request processing is especially needed in meshed SDH networks

There are solutions to automate the traffic rerouting, which are based on predefined restoration scenarios. These solutions, however, need a lot of

implementation effort and consume more resources in terms of traffic capacity.

Summary of the Invention

The present invention proposes a two-phase approach that accelerates the normal hardware re-configuration in a network element.

Each configuration request is divided into two phases: At first the "Fetch-Ahead" phase and then the "Consolidation". During Fetch-Ahead no consistency checks are performed, the changes are not made persistent in the database and only the absolutely necessary changes are done. After Fetch-Ahead these changes are temporarily valid in all SW layers. During Consolidation all the remaining things, which are skipped during Fetch-Ahead, are done.

This accelerates re-configuration, especially in restoration scenarios. If two requests follow each other directly, the second request is delayed until the consolidation of the first one has been finished.

Detailed Description of the Invention

In the following, preferred embodiments of the invention will be described in more details.

CHAPTER 1: Introduction

1.1 Purpose of the Document

The purpose of this document is: 1

- To provide a detailed external description (behavior) of this feature as a whole. 2
- To provide an overview of the general and specific requirements for the feature. 3
- To specify the impact of this feature as it is implemented across the various segments. 4
- To describe new or changed interfaces between all affected segments of the DXC software. 5
- To identify the overall test strategy to be applied for the feature. 6
- To identify all assumptions, limitations, and risks associated with the proposed design. 7

This document should be used as the reference document for more detailed design of the affected segments. 8

1.2 Scope of the Document

This document describes the behavior and implementation of the Fast Restoration feature as a whole. The feature implementation is discussed, including – but not limited to – its general capabilities, the external behavior as represented by the user interface, network interfaces, hardware interfaces, services, and limitations. Also described is the impact of the feature on existing segments with emphasis on any changes in the current behavior or functionality. Fast Restoration will be first implemented in 1674 LamdaGate 1.3. It is planned to support also for R2.3 Fast Restoration. 9

1.3 Organization of the Document

This section gives a brief overview of how this document is organized. It is important to understand the organization of the document in order to follow the contents in an easy way. 10

- **Chapter 1** provides an introduction to the document by describing the purpose, scope and organization of the document. 11
- **Chapter 2** provides a description of the feature's external behavior in such a way that it is clear to any customer, test engineer or design engineer what capability is provided and how it works. This can be illustrated by well-chosen scenarios. The chapter describes also all requirements. 12
- **Chapter 3** describes the changes to segments required for the implementation of the Fast Restoration feature. 13
- **Chapter 4** identifies the impact on existing or new interfaces. 14

- **Chapter 5** provides basic scenarios 15
- **Chapter 6** identifies all assumptions on which the design is based. In addition, any li- 16
mitations imposed by the design are clearly noted. Finally, any risks or undesirable side affects which can be foreseen are described.
- The **Glossary** and **List of Abbreviations and Acronyms** sections define the terms and 17
abbreviations used in this Concept Paper.
- **Bibliography** provides a list of documents referred to within this Concept Paper. 18
- **History** lists the Change History of the Concept Paper and the Planned Changes for the 19
next edition of the Concept Paper.

5

CHAPTER 2: Feature Description

This chapter provides the user with a complete understanding about how the Fast Restoration feature works from an external point of view. 20

2.1 Background

Flexibility, efficiency and restoration make meshed networks getting more and more interesting as alternative to ring network structures. Today the decisive disadvantage of meshed networks is the long restoration time. Lab tests have shown that in a quite simple network topology with 3 nodes the restoration of 8 AU4−4c takes about 10 seconds. This time is not acceptable for the customer since other vendors offer restoration within 200ms in such simple networks. 21

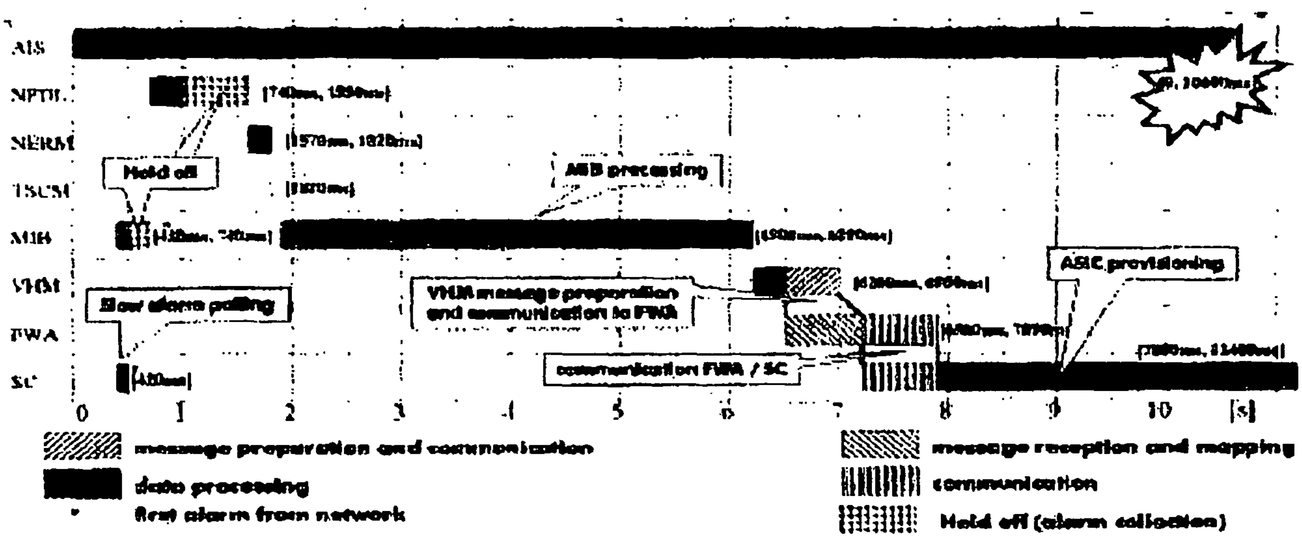

Figure 1: Actual Situation

A detailed analysis, see also Figure 1, has shown that the main time slice is consumed in the NE for the implementation of the new paths. A less significant but nevertheless not negligible delay is caused by alarm detection and propagation. Further delays are intentionally implemented in MIB/NP to cumulate alarms before reporting / reacting. 22

It has to be mentioned that the current SW architecture was implemented with an emphasis on a secure provisioning mechanism and naturally this security is paid with additional processing time. 23

To be competitive a restoration time of 2 seconds and less has to be achieved when restoring 64 single VC4. The proposed solution tries to keep the benefits of the current implementation and the general architecture of the DXC while accelerating the path implementation. 24

2.2 General Description

Two phases have to be distinguished for the improvements: 25

- alarm detection, alarm propagation and path rerouting 26
- path implementation 27

The first phase needs minor architectural changes and optimizes the actual implementation 28
and each involved component. At first the alarm detection time at SC level is reduced for important alarms (LOS, LOF, MS–AIS). These alarms are not delayed with a hold off time in MIB or NP but they are immediately processed. Also secondary path alarms, mainly VC–AIS or VC–SSF are polled with a higher frequency (but not as fast as LOS, MS–AIS). As a consequence the hold off times in MIB and NP can be reduced because the alarms from all I/O subsystems in the DXC, respective all network elements in the network, should be received in a smaller time window.

Receiving a primary alarm NP will analyze which VC4 paths are affected and then it starts 29
immediately the rerouting.

The major objective for path implementation is performance. Not less important is that the 30
system reliability is not affected. To meet these opposed requirements it is necessary to split the processing in the NE into two steps:

- The first step (fetch–ahead) is designed to minimize processing time and to implement the 31
  new paths as early as possible. This step provides reduced security against process restarts and ignores all activities which are not absolutely necessary for path restoration.
- The following second step (consolidation) is executed in the traditional way (slower & se- 32
  cure). It is very similar to the actual implementation.

In nearly all cases the fetch–ahead will be successfully executed and only if an exceptional 33
event happens, e.g. a process crashes, it may fail. Such an exception will only affect the path restoration time since the following slower consolidation step will anyway create the requested new paths.

An overview scenario is given in Figure 2. The MIB expands the compact message format of 34
the restoration request, scans the request and filters all connection relevant information. The filtered information is directly translated into the VHM message set and sent to VHM (Msg 2).

7

ALCATEL

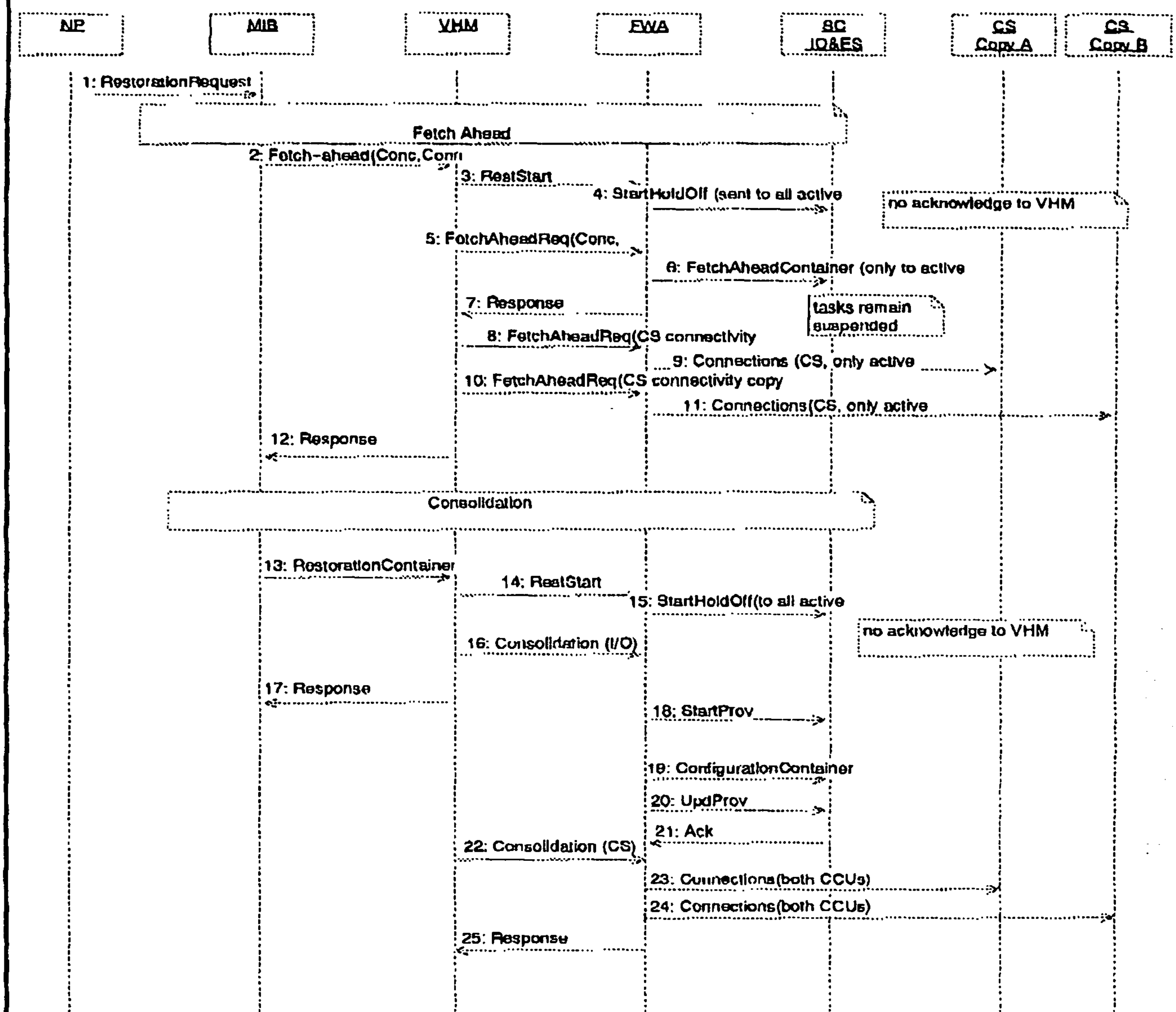

Figure 2: Overall Scenario Path set up

As today VHM/FWA broadcasts the message 'startHoldOff' to all I/O shelves so that no connection failed alarm nor consequent actions nor matrix EPS is executed while the connections are instable. Then VHM disables the persistency management, e.g. all the following modifications are not written to disk or copied to the redundant NAU but they remain only in memory. The information received by VHM is already filtered for connection relevant information. As a consequence VHM activity is automatically focused on creating the new paths. Further improvements, e.g. special processing for the fetch–ahead, or the collection of concatenation information per shelf is implemented to reduce processing and communication times. When all configuration is calculated VHM sends the configuration messages to FWA. Also these messages contain the 'fetch–ahead' indication. 35

The center stage configuration is sent as a normal connection request which contains the 'fetch–ahead' indication. The configuration messages are duplicated by VHM to both center stage matrix copies. 36

3007 A4

For the SC configuration FWA receives only a single message per I/O shelf which contains 37
the concatenation and endstage connection information. This information is sent as a single container message to the active SC and implicitly includes 'StartUpdProv' and 'UpdProv'.

SC analyses the database changes and limits the calculation of register maps to connection 38
relevant information. This will partly be achieved because FWA sends a reduced number of blocks to the SC. Nevertheless SC executes an optimized ASIC provisioning because it knows that this fetch–ahead pass is limited to the creation of new connections.

As a result the paths are very early restored. But the fetch–ahead leads to an inconsistent con- 39
figuration which will not survive any process restart or EPS. The configuration is inconsistent with respect to PM, path monitoring, internal supervision configuration. Due to the inconsistency the IO shelf SC will not resume the suspended tasks – this is done when the consolidation is received.

When VHM and FWA have finished their communication VHM sends the fetch–ahead re- 40
sponse to the MIB and then MIB starts processing the original restoration request in the normal way (consolidation): It creates and updates all MOs, connections, etc. and makes them persistent. At the end the consolidation request is sent to VHM.

Even for the consolidation the startHoldOff has to be broadcasted to all I/O shelves. This is 41
necessary for certain SC restart scenarios.

Also VHM manages this request as today: all connections, monitors, etc. are configured and 42
then persistency management is enabled again. When all data is persistent MIB gets the acknowledge for the request. VHM shall reuse results from the fetch–ahead step, mainly the matrix routing must be the same in order to avoid hits on already restored traffic.

During consolidation FWA maps all configuration blocks and sends them to the SC. Some infor- 43
mation from the fetch–ahead will be simply re–written and there is no need to optimize the mapping.

The SC processes the consolidation request and afterwards all tasks are resumed. 44

2.2.1 Overlapping Restoration Requests

This chapter has still to be completed but for the implementation of the first iteration it should 45
be considered that the system has to managed a second restoration requests before the consolidation of the first request is completed. This is not planned for the moment.

CHAPTER 3: Requirements

The requirements for restoration are described with Figure 3. In case of a fiber cut for the normal traffic between LG1 and LG2 the traffic has to be restored within 2000ms. 46

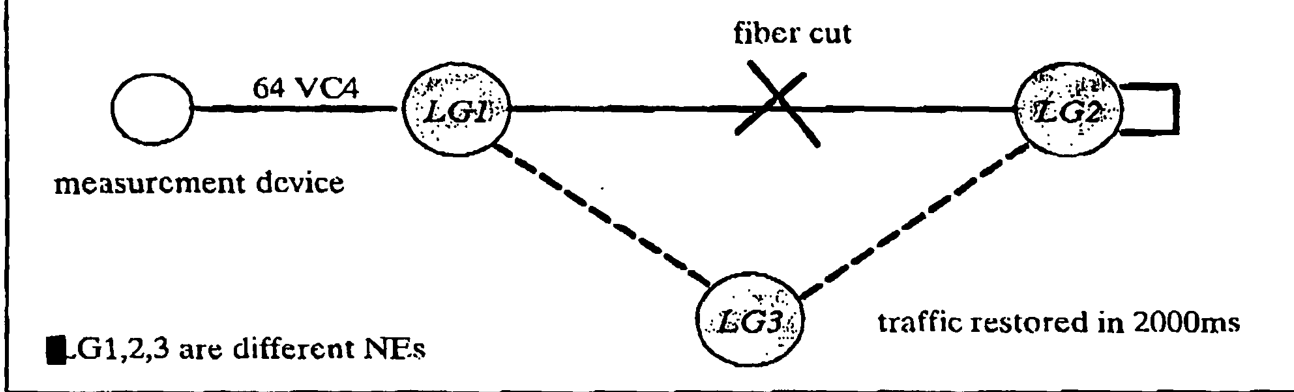

Figure 3: Reference Testenvironment

The overall requirement of 2000ms protection time is detailled with Figure 4. It defines the maximal times the different processing steps may need. 47

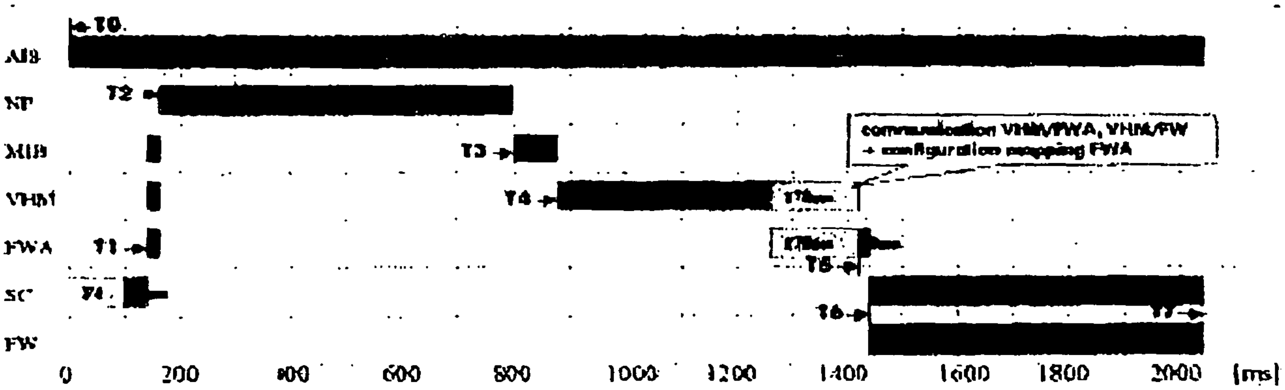

Figure 4: Target scenario with 2000ms restoration time (Fetch−ahead)

- FWA shall receive the primary fault at T1=150ms (F4 filter value is 100ms) 48

  NP shall receive the primary fault at T2=160ms 49
- At T3=800ms MIB shall receive the restoration request. 50

  100ms later, at T4=890ms VHM receives the filtered Fetch−ahead configuration 51

  I/O respective matrix subsystems receive the last configuration message at T6=1440ms. 52
- ASIC configuration is finished at T7=2040ms. 53

CHAPTER 4: Architecture and High–Level Design

This chapter describes the architectural impacts of the segments. Figure 5 gives an overview about the processes involved in restoration (with the focus on fetch–ahead) and how they interwork. In the following subchapters they are described in more detail. 54

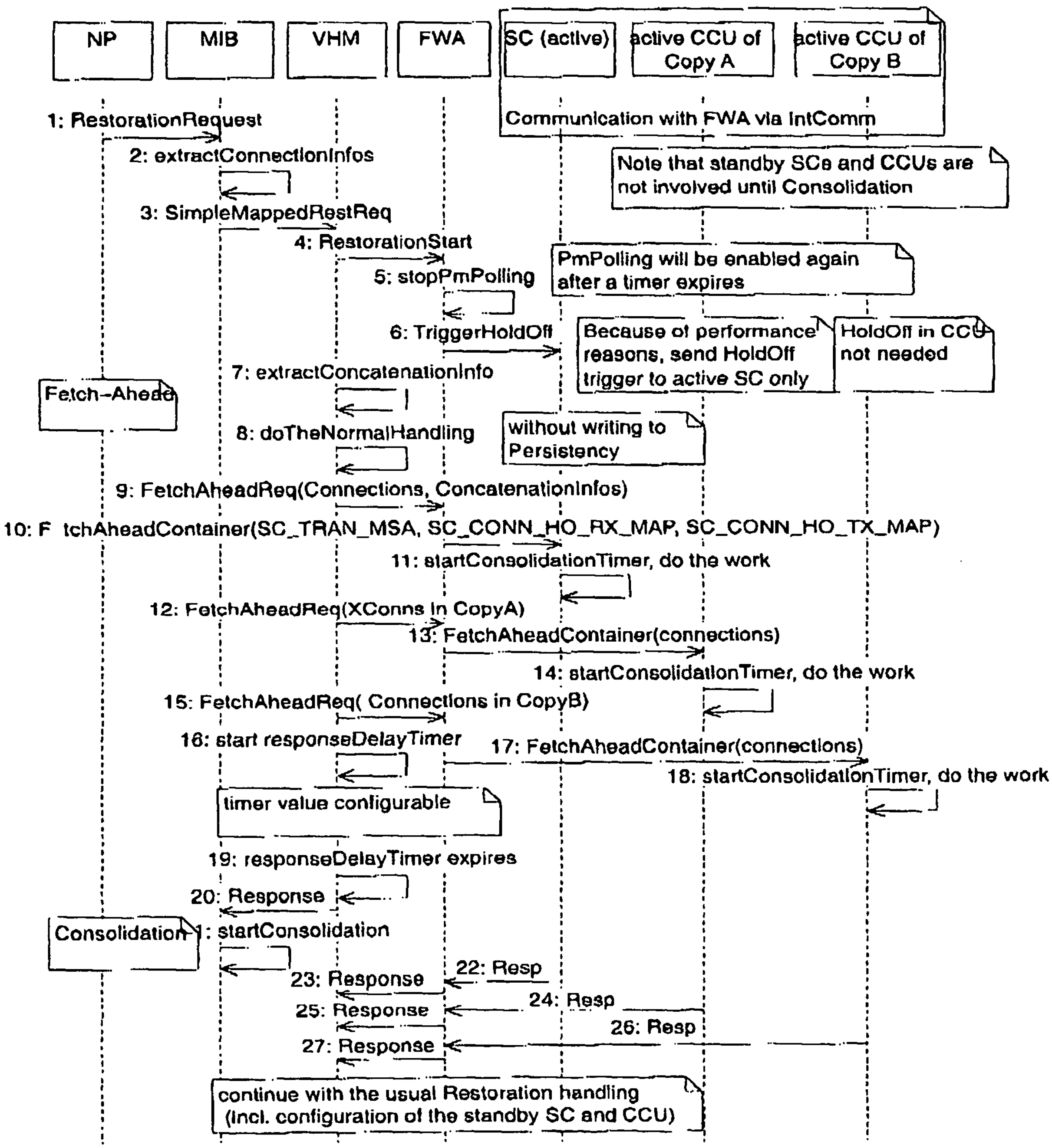

Figure 5: Fast Restoration Overview

4.1 MIB

Three main changes are necessary in the MIB: the alarm handling, the two phase approach 55
with fetch–ahead and consolidation and improvements during consolidation.

4.1.1 Alarm Handling

At the moment, received alarm notifications from VHM are stored in MIB until a hold–off 56
time of 300 msec expires. This is no longer feasible for fast restoration. It must be changed in the following way:

- Incoming alarms are collected in a *ChangesCollectionReport* 57
- Check the *ProbableCause* of the alarm 58
- If the *ProbableCause* indicates, that it is a primary alarm (LOS, LOF, MS–AIS), this alarm 59
  and all alarms already collected in the *ChangesCollectionReport* are sent to NP at once
- For all other alarms a configurable hold–off timer (range: 0..300 msec) is started (if it is the 60
  first alarm in *ChangesCollectionReport*)
- On expiration of this hold–off timer or if a primary alarm arrives (see above) all collected 61
  alarms are sent to NP

A further needed improvement is to send the alarms to NP **before** they are stored in Persistency. 62
At the moment the alarms are sent in the *committed* phase of a transaction. In future they must be sent in the *prepareToCommit* phase. This ensures that the writing to Persistency and especially the sending and writing to the redundant AU is done after sending the message.

4.1.2 Two Phase Approach

A new mechanism is needed for the fetch–ahead processing, i.e. the mapping of the incoming 63
request to a VHM request and nothing else, especially no storing to Persistency.

In addition it must be ensured, that after every fetch–ahead processing the VhiSendController 64
sends at once the resulting message to VHM and that an eventually incoming further request (restoration or non–restoration) is not handled until the consolidation phase is finished.

To avoid, that the processing of the restoration request is delayed in situations where a lot of 65
requests from RM or a lot of alarm events from VHM are queued in MIB, the handling of the requests coming from NP must be prioritized (☞ 4.2).

The restoration request must be filtered for connection relevant data to be mapped and sent 66
directly to VHM. These data are:

- connections (both for creating/activating and for deleting/deactivating requests). Note that 67
  commands e.g. force switch in case of an SNCP are ignored in VHM during fetch–ahead (but are needed during consolidation e.g. because of soft rerouting)
- concatenation levels 68

  In fact, MIB sends the connection requests towards VHM and VHM is responsible to pro- 69
  vide the correct concatenation level of the TPs to be connected, i.e. MIB does not send an explicit concatenation request.

Before starting the consolidation, MIB should wait some time in order to give VHM, FWA 70
and IntComm processes access to the system resources, especially the CPU computing time.

Therefore for the fetch–ahead communication an own socket is used between MIB and VHM (this socket is prioritized in VHM). When VHI has sent the fetch–ahead request to VHM, it waits until VHM sends the fetch–ahead response on this socket (☞ Figure 5). This response indicates that all lower level processes have completed their fetch–ahead processing.

The waiting on this special socket should be implemented by using the *progressInstance* method 71
of SocketComm. This ensures that no other requests are handled. Because *progressInstance* returns not only when a user message arrives but also on arriving of a *alive* message (even if this *alive* message is received on another socket!), on every return MIB has to check whether the *response* has arrived from VHM. Then the consolidation can be started at once.

To handle errors, where VHM never sends the *response*, MIB needs a time–out. Unfortunately 72
it is not possible to do a *progressInstance* with time–out. Therefore the alive mechanism is used instead: The fetch–ahead socket is instantiated with a keep–alive of 30 seconds (a long time–out but it is only needed in a very improbable error case). This ensures that at least every 30 seconds the *progressInstance* returns. Because it returns too when an alive message is received on another socket, the time the request was sent, must be kept in MIB (not persistent). The waiting for VHM is stopped if more than 10 seconds have been passed. Even in this error case, the consolidation is started (including the normal error handling, if needed).

In case of a VHM process restart the fetch–ahead socket is closed. This is a trigger for MIB 73
to start the consolidation at once. Closing the fetch–ahead socket means that the method *connBroken* is called. Furthermore the *progressInstance* method returns at once with return code *false*.

Special care has to be taken for cases where the consolidation phase aborts for any reason (e.g. 74
MIB process restart or if the restoration request is rejected by MIB during the checks in the consolidation phase). Because no storing to Persistency is done in the fetch–ahead phase (even not the request itself), in case of a MIB process restart, the restoration request is totally lost in MIB. The NP detects this MIB restart and starts at once a rerouting over other NEs. But because the fetch–ahead request is already sent to VHM and therefore also valid in the HW, a mechanism is needed to trigger a fall back to the former state: As soon VHM detects that the fetch–ahead connection to MIB is broken, it has to reestablish its state before the fetch–ahead request (by a simple process restart or an explicit transaction rollback). Also the SC and the CCU must fall back to the configuration before the fetch–ahead (see below for details).

If MIB rejects a restoration request, the VHM (and again SC and CCU) must also reestablish 75
their pre–fetch–ahead state. MIB can trigger this by a simple close and restart of the fetch–ahead connection to VHM. This is interpreted in VHM like a MIB process restart and handled accordingly.

Note that as long as NP does not distinguish between Restoration, Soft–Rerouting and normal 76
connection build up, every (restoration–) request is handled in two phases, i.e. in total more time is needed between request from NP and response to it (after consolidation phase is finished).

4.1.3 Improvements during Consolidation

Though the fetch–ahead ensures, that the disturbed traffic is very fast restored, the consolida- 77
tion itself must be improved: Find the weak points e.g. with the tool Quantify and solve the already known problems (☞ SLSks25202).

13

4.2 MIB Framework

To ensure that a request from NP is always handled with the highest priority (consider a situation 78
where a lot of requests from RM or a lot of alarm events from VHM are queued in MIB to be processed), the priority of the Session Request Handlers, especially of the user NP, must be configurable (see following figure and description).

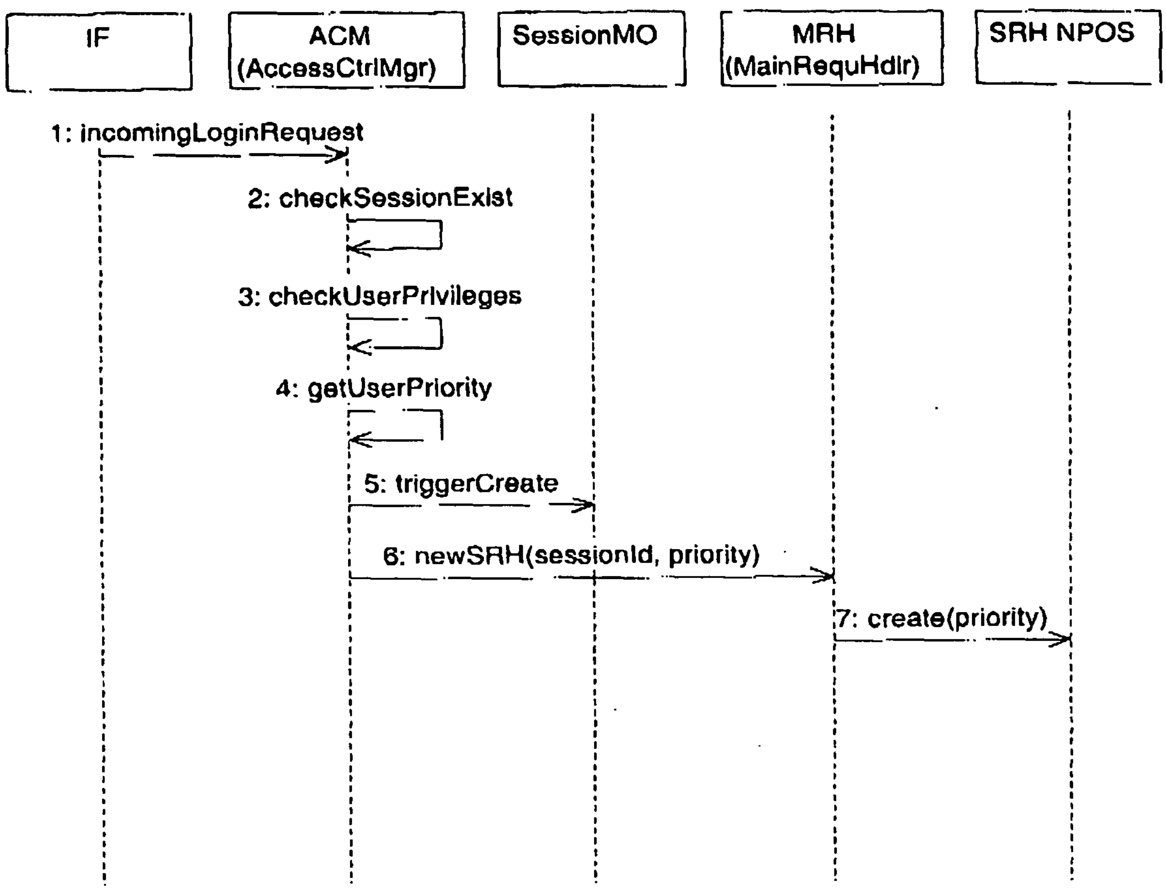

Figure 6: Prioritization of SRH

When a user tries to log–in, the ACM has to check the authorization and privileges of this 79
user. These privileges must also determine a priority class. The value itself can be determined per configuration file like it is already done for the type SRH. The priority (or the priority class) then can be passed to the MRH which creates the SRH NPOS (and implicitly the corresponding phase) with the higher priority.

4.3 VHM

To implement the two pass approach, VHM has to do: 80

4.3.1 Handling and forwarding of fetch–ahead indication

A new message attribute has to be defined in the VHM message set, indicating the fetch–ahead 81
(to indicate the consolidation messages, the Restoration Container message is used). This

90071 A4

fetch–ahead request message is sent from MIB via a special socket connection with a higher priority than the normal connections to ensure the precedence of the fetch–ahead.

When receiving the fetch–ahead request, VHM has to change its internal state to indicate 82
that a special handling (see below) is needed. Beside of this, VHM sends at once (as usual) the RestorationStart message to FWA, adapts the concatenation levels of the involved TPs (☞ 4.3.4) and performs the usual connection handling.

At the end of the fetch–ahead processing, VHM sends the new connectivity messages together 83
with the concatenation information in a single message per shelf (☞ 4.3.5) with a fetch–ahead indication to FWA.

4.3.2 Inhibit Persistency during fetch–ahead

To avoid the time consuming writing to Persistency during fetch–ahead, VHM disables Persis- 84
tency. This can be done centralized in the VHM platform: The during fetch–ahead changed objects register themselves as usual in ObjCommDB resp. PersRecHdlr. But at the end of the request processing they are not written to Persistency but are still kept registered. This ensures, that they will be written after the following consolidation processing, though they may not (again) change their attributes during consolidation (and therefore usually will not register themselves).

Open issue: PersRecHdlr receives and keeps only pointers to the data to be stored. Therefore 85
the databuffers behind these pointers must not be deleted until consolidation.

In addition the handling of empty transactions should be improved: Avoid the starting of a 86
Persistency transaction if no data has to be stored, eg when receiving an intermediate response from FW/FWA (☞ SLSks25208).

4.3.3 Limited configuration in fetch–ahead

It has to be ensured that only the messages concerning connections and concatenation are sent 87
to FWA, e.g. no PM configuration (default when changing the concatenation level), no TP configuration, etc.

Normally every changed object registers itself at VHM Flow Control and at the end of the re- 88
quest processing these registered objects are called back to enable them to send their complete configuration to FWA (including the changed attributes). During fetch–ahead all registered objects must be kept registered. In the following consolidation they are called back and they send their then actual complete configuration to FWA (including further during consolidation changed attributes).

4.3.4 Implicit concatenation requests in fetch–ahead

If for a restoration request the concatenation level of a TP has to be changed, VHM has to 89
detect this autonomously in the connection request and must change it. During fetch–ahead, MIB performs a simple mapping without checking the actual states of the involved TPs. Therefore no explicit concatenation request is sent to VHM.

4.3.5 Collect and send concatenation information separately

If the concatenation level changes, VHM has to create resp. delete the corresponding objects. 90
This must be done even during fetch–ahead to allow the creation of new cross connections

via these concatenated TPs (e.g. to be used in the matrix routing algorithm). Usually these new objects send their (default) configuration to FWA, including a lot of information not needed during fetch-ahead.

To avoid these configuration messages, which must be transmitted to FWA, and also to simplify and accelerate the job of FWA, the concatenation information must be stored separately per I/O shelf. This information then can be sent in a new (to be defined) submessage to FWA, together with the connection information. I.e. only one message per shelf is sent (saves communication time) and it is addressed to the active SC. 91

It is a good idea, to store the additional concatenation information at the HCSMX board object because then in the flow control send pass only this board needs to be called back. All other link and I/O boards, as well as the TPs on them, have nothing to send resp. must not send anything. 92

Note that the concatenation information is only needed in the I/O shelves. For the CS shelves the connection information is sufficient. 93

4.3.6 Special socket for fetch-ahead requests from MIB

To prioritize the fetch-ahead request over other actual queued requests (from MIB as well as from FWA, especially PM reports) and also to enable MIB suspending itself (☞ 4.1), a new socket has to be defined. This new socket is exclusively used for fetch-ahead communication between MIB and VHM: MIB sends the restoration request, VHM sends a response (after all messages to FWA are sent and an additional, configurable time-out is expired, ☞ 4.3.7). Its priority is set to *high*, i.e. the same priority as e.g. the configuration socket to FWA. 94

4.3.7 Delay fetch-ahead response to MIB

To give the MIB a trigger, when it can start the consolidation (☞ Figure 5, ☞ 4.1), the fetch-ahead response to the MIB request is delayed until the complete fetch-ahead configuration (for all involved I/O and CS shelves) is sent to FWA. Because FWA then needs some time to map these requests and to send them further on towards SCs resp. CCUs (via IntComm), VHM calls the *msecSleep* method (☞ SEGM/basicfrw/BasicS/UNIXIF/msecSleep.h) after having sent the last request to FWA. The value of this sleep must be configurable. Its range should be from 0 to 500 msec with a default of 50 msec. 95

When this timer expires, VHM sends the response to MIB which starts at once the consolidation. 96

Note: The following discussed alternative to the delay timer is not feasible: For every fetch-ahead request from VHM, FWA sends a notification event back to VHM after it has mapped and sent this request further on to the SC resp. CCU (in addition to the normal response, which is sent after FWA has received the response from SC/CCU). VHM collects all these notification events and as soon as for every involved shelf such a notification has been received, the fetch-ahead response can be sent to MIB. 97

This alternative has the big disadvantage (beside of the bigger implementation effort), that FWA has to do more work during fetch-ahead. Sending a notification event needs an additional phase in FWA and this will delay the execution of the following fetch-ahead requests (for further shelves). 98

4.3.8 Make all changes persistent in consolidation pass

Because in the fetch–ahead pass no storing to persistency is done, all changed objects must 99
be written in the consolidation pass, together with the additional objects, changed during consolidation. Special care has to be taken for data of the MCL category, because here only pointers to data buffers are registered to be stored. It has to be ensured that after the fetch–ahead transaction, these pointers, as well as the related data buffers, are still valid and can be stored in the consolidation pass.

4.3.9 Repeat fetch–ahead configuration in consolidation pass

All the configuration and connection requests during fetch–ahead are sent to the active SC 100
resp. the two active CCUs (of copy A and copy B) only. Therefore they must be repeated during consolidation. For new or changed objects like TPs this can be easily achieved by keeping them registered at the VHM flow control. Then they are called back during consolidation and send their now actual configuration (including further changes made during consolidation) towards FWA.

For connections it is a little bit more complex: The connection requests which are repeated 101
from MIB during consolidation are recognized as already valid. Therefore nothing has to be done in VHM and nothing would be sent to FWA. This ensures that no new routes through the matrix are calculated for the connection requests, repeated in the consolidation.

To ensure that the connection requests are repeated, a total connection reload is triggered, 102
i.e. to all matrix boards in CS and ES the complete list of their active connections are sent.

4.3.10 Delay rearrangement after consolidation

If during a fetch–ahead, the matrix SW recognizes a need for a rearrangement, this part of 103
the connection request is ignored. But all the other possible connection requests are handled as usual in the fetch–ahead. During the following consolidation phase, the same happens: the possible connections are recognized as already present, those leading to rearrangement are negative responded to MIB and the consolidation phase is finished.

Having received a negative response with rearrangement indication, MIB resends these con- 104
nection requests but now outside of restoration. Now they are handled in VHM as usual, i.e. the rearrangement is performed.

4.3.11 Maintenance processes

It is a useless relict from old releases that in 1674LG VHM maintenance processes like CCX, 105
AM and UAEM are running though they are no longer involved in the normal request processing. Though the additional notifications, the VHM process sends to these processes need nearly no time, the problem is that they need system resources (e.g. CPU time). Therefore they must not be started any more (simply change the SSU startfile).

4.3.12 Improve alarm processing

At the moment an incoming alarm is processed by the responsible object(s). Then the persistent 106
data of these objects are handed over to Persistency. Then the alarm notification is forwarded

to MIB and at last Persistency writes these data onto the disk. Improve this handling by sending the alarm notification(s) to MIB before the persistent data are handed over to Persistency.

4.3.13 Improve restoration request processing

The two pass approach with fetch–ahead ensures that the connections are very fast up. But 107
beside of this new mechanism it is useful to check whether further improvements are possible, e.g. avoid unnecessary checks, object creations, comparisons, whatever else. Especially the parts which are executed during fetch–ahead (concatenation, cross connection handling, matrix routing, etc.) must be considered. Tools like Quantify can be used to identify the time consuming parts.

4.3.14 Precedence of configuration

The SCs and CCUs are separated machines and executes the requests in parallel. Therefore 108
the send ordering of the requests should be in a way: the slowest entity first. At the moment the slowest entity in a Square configuration is the Main Shelf SC, in a CLOS configuration it is expected that the I/O shelves are slower than the CS CCUs (has to be verified).

4.3.15 Improve message preparation time

Measurements showed, that the time needed to build up a message to matrix boards (CX1000 109
as well as HCSMX) is much longer than to I/O boards (though the total amount of data is equal). This should be examined and improved.

The configuration for both matrix copies is equal. Therefore the message to the second copy 110
should be simply copied from the first request and not set up from scratch.

4.3.16 CX1000 board configuration

Up to now connections are brought onto the CS matrix boards by triggering a total configuration 111
reload. But during fetch–ahead the FW on the CS CCU does not check its database whether the requested connection is already present. This implies, that all requested connections are brought onto the HW. And because this needs a lot of time, it must be avoided. This is why during fetch–ahead only the new connections have to be sent (note that VHM needs a little more time to setup a list of single connections compared with a simple total configuration reload).

4.3.17 Alarm events between fetch–ahead and consolidation

Though the alarm reporting is switched–off in the I/O shelves by the restoration start message, 112
there may be an event already in the queue. This alarm cannot be made persistent until end of the consolidation phase. Therefore it must not be acknowledged until then. But anyway it can be processed and also forwarded to MIB (which does not handle it until the end of consolidation).

In CS the alarm reporting is not switched–off, but the probability of any alarms is very low. 113
Anyway any eventual alarm can be handled like described above.

Alternatively VHM can just ignore the incoming alarms. Flow control mechanisms in FWA and SC resp. ensures that this alarm is repeated later. 114

4.3.18 EPS switch of SC between fetch–ahead and consolidation

If for any reason an SC or CCU performs an EPS switch between fetch–ahead and consolidation, it will lose the configuration sent in the fetch–ahead request, e.g. new connections are removed as soon as the former standby SC resp. CCU is up. Because of the small probability of an EPS switch (note that the hold–off time is active), it can be accepted that here the connection request is delayed until consolidation. 115

If VHM receives the switch indication from an SC resp. CCU, it must keep this in mind and at the end of the consolidation this switch indication triggers a complete configuration reload of this shelf including the changes for the restoration request, handled in the consolidation. 116

4.3.19 Restart of MIB between fetch–ahead and consolidation

A process restart of the MIB can be detected by VHM because of the broken fetch–ahead socket communication. Getting this trigger, VHM has to reestablish its state before the fetch–ahead because MIB has no persistent knowledge of the restoration request until the end of its consolidation phase. 117

To reestablish the former state, VHM can either perform a process restart too (quick and dirty implementation but needs more processing time) or can perform a transaction rollback. In both cases it loses the fetch–ahead state indication. 118

Some time after, on SC resp. CCU the consolidation time–out exceeds. Then the SC resp. CCU performs a reload out from their former database contents (SC by a warmstart, CCU a simple reload from its database). Note that CCU does not write the changes for fetch–ahead in its database while SC has two separate databases (one for the last fetch–ahead changes, one for the state before). After this reload, also in the HW the pre–fetch–ahead state is reestablished (without a need for VHM to send again the complete configuration). 119

4.3.20 Rejected restoration requests

Because MIB performs no checks in the fetch–ahead, it may happen, that the fetch–ahead request is already performed in VHM and already present on HW but MIB rejects it during consolidation. In this case a reestablish of the pre–fetch–ahead state in VHM and HW is necessary. MIB can trigger this by simply closing the fetch–ahead socket to VHM. VHM will interpret this like a MIB restart. For the following actions see 4.3.19. 120

4.3.21 Normal request between fetch–ahead and consolidation

This is an error case which should not happen: MIB performs no other request between fetch–ahead and consolidation. Anyway because of security reasons VHM should handle this case: When receiving a not consolidation request after fetch–ahead, VHM should reestablish its former state like described in 4.3.19. The incoming normal request is ignored and therefore repeated later by MIB. 121

4.3.22 Restart of VHM between fetch–ahead and consolidation

A restart of VHM does not matter (of course MIB must recognize this to stop the waiting for the fetch–ahead response, ☞ 4.1) because MIB will send the complete restoration request 122

during consolidation. The only thing is: VHM must not reject a consolidation without a previous fetch–ahead.

Furthermore it is important that VHM sends again the RestorationStart message to trigger 123
the hold–off in the shelves. Note that the SCs may have already performed a warmstart because of consolidation time–out.

4.3.23 Improve handling of PM History reports

If a fetch–ahead request arrives at VHM directly after a PM history report, the restoration 124
request will be delayed for the time needed to store the complete *potential* history data of this board. Though no lower order TPs are possible in 1674LG, storing and especially transmitting the PM data of path and tandem connection monitors before and behind matrix needs some time.

As long as this transaction is ongoing in VHM, the sockets can not be checked. Therefore this 125
potential delay must be accepted but note that higher priority of the fetch–ahead socket ensures that if both a PM report and a fetch–ahead request is waiting at the same time, the fetch–ahead request will be performed first.

4.3.24 Impacts and Caveats

Note the following impacts: 126

- If a consolidation request contains less or other crossconnections than the former fetch– 127
  ahead request we get an inconsistency between MIB and VHM/hardware
- VHM sends no SNCP commands towards FWA in fetch–ahead phase 128
- After having processed the fetch–ahead request the VHM internal flow control callback to 129
  send FW requests is disabled until a new fetch–ahead request or a consolidation request arrives

4.4 FWA

As today FWA has to broadcast the startHoldOff message to all SCs. when it receives the RestStart message. But in future FWA will not acknowledge the RestStart message. 130

With a single message FWA receives the changes of the connectivity map for the end stage modules and also the changed concatenation information for all ports in the shelf. A special mapping procedure limits processing to this information and generates a single fetch–ahead container message to the active SC. This message contains only configuration changes for connectivity and concatenation. The message doesn't need a startProv / UpdateProv sequence. 131

As a second improvement FWA shall inhibit any PM history handling for N seconds as soon as it receives the RestStart message at the beginning of the restoration. The PM history handling is continued when the timer expires. This is also necessary because any SC that has received fetch–ahead request has suspended its tasks and cannot deliver PM history data. The timer mustn't be restart save. 132

In case FWA restarts the actual mechanism applies. VHM repeats the configuration after flow control time–out. As a consequence the fetch–ahead may fail and paths are not restored in a fast manner. 133

FWA shall send the center stage configuration to the active CCU only. 134

FWA doesn't receive the syn4CS configuration message during fetch–ahead. It has to create a new mapping function in order to calculate the TX connectivity block. 135

4.5 FW

The CX1000 reload mechanism isn't suitable anymore for large matrices. The new connectivity is sent with a normal connection request containing only changed configuration data. The request contains the fetch–ahead indication and FW shall start a 'consolidation supervision timer'. This timer is cleared as soon as a consolidation configuration is received. 136

The fetch–ahead configuration bypasses the normal process architecure (database, hardware updater, etc.) to improve the performance. It is not necessary to store fetch–ahead information in the database – this will be done with the consolidation request. 137

Only in case the fetch–ahead couldn't be closed with the corresponding consolidation pass (MIB or VHM crashed) the timer expires and FW shall reload the unchanged configuration of the database. 138

In restart scenarios it may happen that a CCU receives a consolidation request without a preceding fetch–ahead. This request must be accepted like a normal configuration request. 139

FW has to distinguish between consolidation configuration and normal configuration requests. 140
If the FW receives a normal request while expecting a consolidation request it shall reload the database into the ASICs before processing the request:
Assume that the FW has received the fetch–ahead configuration and then VHM restarts and processes another configuration (not a consolidation). If the FW would accept this request a mixture of the unconfirmed / invalid fetch–ahead configuration and the normal configuration would be applied to the hardware and the configuration know by VHM and FW were inconsistent.

The FW shall also accept further fetch–ahead configurations before the consolidation arrives. The configuration applied to the hardware is the merged fetch–ahead data. Each fetch–ahead restarts the 'consolidation supervision timer' (25sec). 141

In R2 the FW shall achieve alarm detection and reporting with less than F4 value+50ms for primary faults (LOS, RS–AIS). 142

The fetch–ahead and consolidation is done as for the CCU in R1. 143

4.6 SC

The following chapters give only some ideas how the SC could improve the performance. 144

4.6.1 Polling

Interfaces and Sections shall be polled with 50ms. Path monitors are also polled with a higher frequency but not as fast as interfaces and sections. 145

4.6.2 Fetch–ahead Configuration

During fetch–ahead FWA will only send connection related configuration with a fetch–ahead container to the SC, therefore provisioning activity is already reduced due to the limited number of changed configuration blocks in the SC database. 146

The fetch–ahead container implicitly contains startProv and UpdateProv and is not stored in the SC database. SC will implement the new connections and disables SUT generators. Afterwards all tasks remain suspended until the consolidation request arrives. This abnormal condition for the SC is supervised with a 'consolidation supervision timer'. 147

If the timer expires before the consolidation configuration arrived the SC will execute a warmstart using the unchanged database content (rollback from fetch–ahead). The fetch–ahead configuration which was configured in the ASICs is lost. Afterwards the SC is in normal operational state. 148

Between fetch–ahead and consolidation the SC shall accept further fetch–ahead configurations. The 'consolidation supervision timer' shall be restarted after each fetch–ahead request. 149

If the SC receives a normal configuration request it will immediately execute a warmstart using the old database content. In this case the SC will not generate any answer for the configuration request. For further information see also chapter 4.5. 150

In restart scenarios it may happen that the SC receives a consolidation request without a precedent fetch–ahead. This request must be managed like a normal configuration request. 151

An estimated list of ASICs and the information that has to be partly reconfigured during Fetch–ahead is listed below (overview): 152

- CRISTALLOS / TIEPOLO (port cards) 153
  new concatenation mode
  (no F3/F4, POM, TTI etc. configuration is changed)
- CRISTALLO (link boards) 154
  new concatenation mode
  (no F3/F4, POM, TTI etc. configuration is changed)

- OTIF Link boards 155
  No reconfiguration on OTIF link boards is necessary: Concatenation is not managed on OTIF and I0/I1 identifier handling is suspended during holdoff. Therefore no EPS, consequent action or connFailed reporting is active.
  Note:
  Upon reception of the 'start hold off message' SC has to disable consequent action AIS insertion due to I0/I1–TIM. This is not impacting the fetch–ahead performance because VHM sends it before starting its own fetch–ahead processing, i.e. the fetch–ahead configuration arrives later.
- EVEREST (both matrix boards) 158
  disabling of SUT
  The new connectivity map is written to the active matrix first although this doesn't lead for all paths in the CLOS matrix to an earlier connection set up. Figure 7 shows that in each I/O system a different a matrix copy may be active. The path from I/O X to I/O Y is only established if also the stand by endstage in I/O X is configured.
  Configuring the active matrix first is the best choice, e.g. I/O shelf Y only the configuration of the active matrix establishes the connection for a signal coming from the center stage. The connection on the standby matrix is not important for this signal direction because the I/O doesn't select the stand by copy.

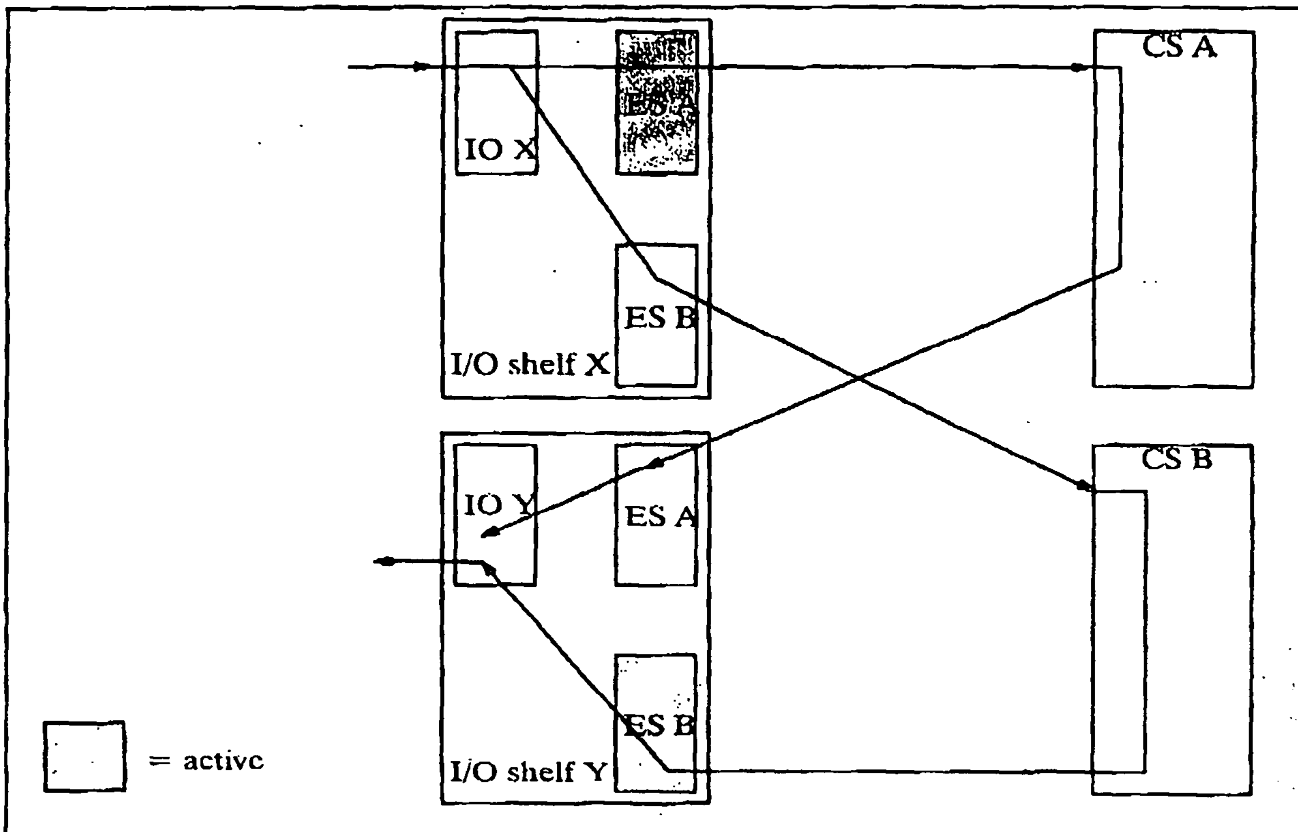

Figure 7: Reference Testenvironment

A general performance improvement could be achieved when 'Pure Quantify' is used to analyse 157
where the SC spends most of the processing time.

CHAPTER 5: Compatibility with other Features

This chapter describes how the new feature is interworking with other DXC features. 158

5.1 Feature Compatibility

- **Matrix rearrangement** 159
  In case a path cannot be routed without rearrangement it is excluded from fetch−ahead and consolidation. As today these paths are created afterwards with normal MIB configuration requests.
- **Start hold before the restoration request** 160
  VHM/FWA have to broadcast the start hold off message to all active SCs. This is done when the fetch−ahead request is received from MIB. SC has to disable also automatic AIS insertion due to I0/I1 TIM on OTIF link boards.
  FWA will not acknowledge the RestStart message.
- **NP management** 161
  NP uses the same restoration requests as in the current systems.
  Also soft rerouting and path setup is managed with restoration requests and they're also split into fetch−ahead and consolidation steps.
  SNCPs used for PRC are not affected, see below.
- **SNCP** 162
  Any SNCP will protect the traffic before the restoration manager calculates a new route. As a second step NP might reroute the failing path to re−establish the SNCP redundancy. This is not time critical and fetch−ahead/ consolidation is compatible. Protection commands are considered during fetch−ahead processing.
- **Soft Rerouting** 163
  see SNCP
- **Alarm reporting** 164
  I/O shelves which receive fetch−ahead configuration suspend their tasks and no alarm reporting is done. When the tasks are resumed the actual alarm states are reported if they differ from the last reported alarm state.
- **Broadcast** 165
  no impact
- **Center stage matrix redundancy** 166
  no impact, also during fetch−ahead both matrix copies are configured.
  In case of SDH matrices they are symmetrically configured. For OTH matrices both copies get different configurations.
- **OTIF link supervisi n and SC EPS** 167
  During fetch−ahead the OTIF related configuration blocks are not sent to the SC. As a result newly created paths do not send I0/I1 identifiers (no sent identifier configured) and they do not expect identifiers (number of expected identifiers is 0). The automatic consequent action AIS insertion due to I0,I1−TIM has to be disabled as soon as the 'StartHold-

Off' message is received. This must be done in all shelves and not only in those that later receive a fetch–ahead request.
Paths which are removed are still expecting identifiers and will detect connFailed. Due to the hold off that was started at the beginning they will not execute consequent actions. The related connFailed alarm is suppressed and no SC EPS is triggered

- **Tandem Connections** 168
  Tandem connections are not reconfigured during fetch–ahead. If a TC source has to be moved to another port this might extend the down time of the path due to consequent actions at the sink.
  No impact for the restoration performance is seen if the TCsource/sink is placed at the NP domain ingress/egress port.
  Note:
  A TC on an intermediate path segment is not really sensible: Any restoration which reroutes the segment will clear TC–PM data because the TC is disabled.
- **Tandem Connections for SPRC** 169
  SPRC based on TCs can only be used if they cover the complete restoration domain, see Figure 8.
  In case A) SPRC and restoration are congruent and a failure will be protected in a fast manner. To re–establish the SPRC redundancy NP will reroute the failing working traffic without reconfigurung the tandem connection, i.e. even if during fetch–ahead the TC configuration is not considered the path down time is not affected.

  In case B) SPRC and restoration domains are not congruent and a defect at the indicated position, outside the SPRC and inside the restoration domain would require to move the TC termination on the left NE to the protection port. As long as the TC is not configured the TC sink at the right would detect TC–UNEQ and the path is forced to AIS.

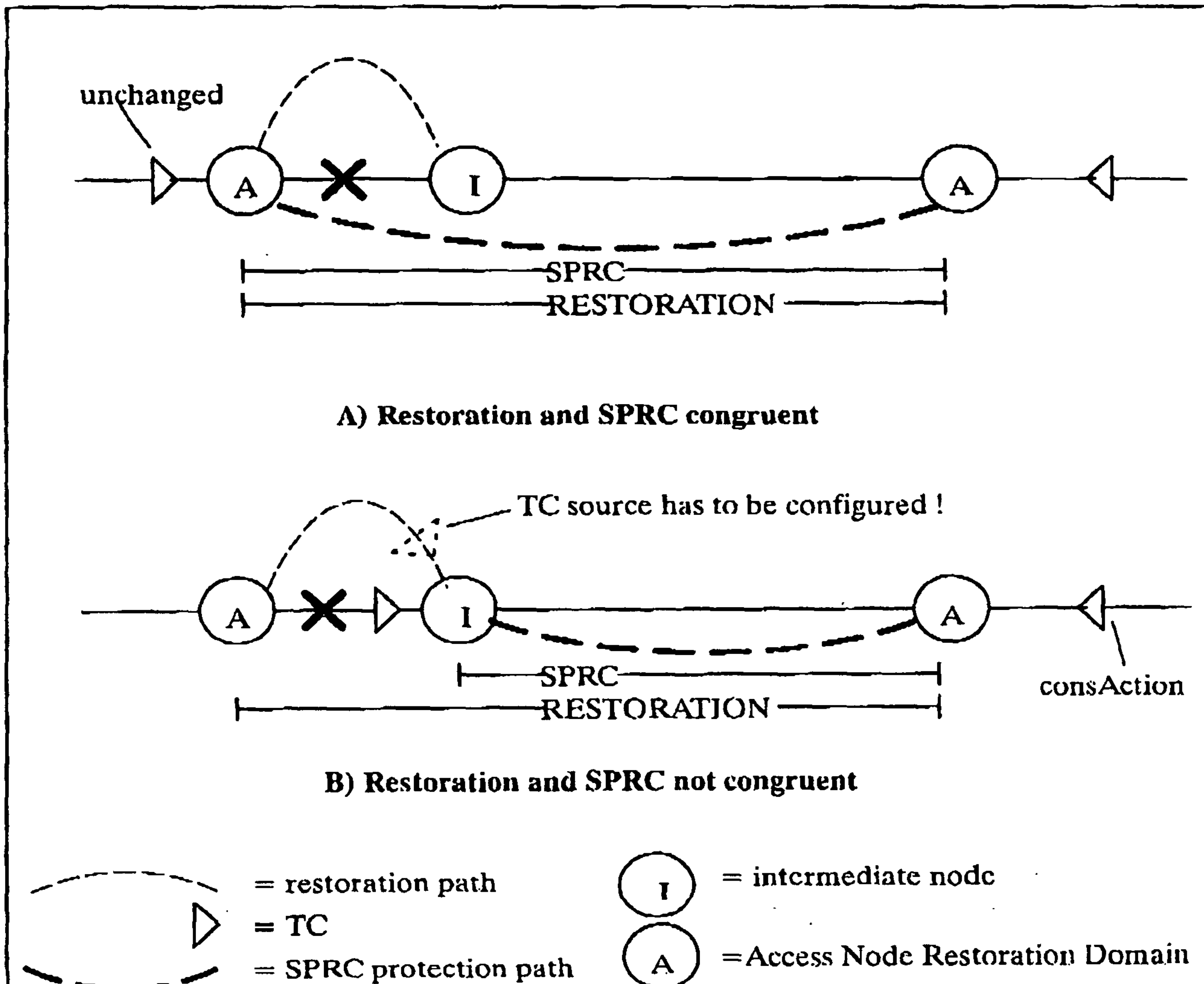

Figure 8: Fast Restoration and SPRC

- **Performance monitoring / getting history data** 170
  Perfomance monitoring is not reconfigured during fetch ahead.
  PM history data handling must be suspended during restoration activities to improve performance in the NAU and because the tasks in the SC are suspended.
- **SC EPS** 171
  If SC executes an EPS before the consolidation is finished it will loose all fetch–ahead configuration and the time needed for restoration is extended for the involved paths.
  No EPS will be triggered by connection supervision because of the running hold–off.
- **MSP** 172
  MSPs are not managed by NP but they might have been locally created at the NE – even for ports that are used for restoration. The NE SW layer have to consider this.
- **MSSPRING** 173
  n.a. for restoration